

The Epistemic Representation of Information Flow Security in Probabilistic Systems

Paul F. Syverson

Center for High Assurance Computing Systems
Naval Research Laboratory
Washington, DC 20375

James W. Gray, III*

Department of Computer Science
Hong Kong University of Science and Technology
Clear Water Bay, Kowloon, HONG KONG

Abstract

We set out a logic for reasoning about multilevel security of probabilistic systems. This logic includes modalities for time, knowledge, and probability. In earlier work we gave syntactic definitions of multilevel security and showed that their semantic interpretations are equivalent to independently motivated information-theoretic definitions. This paper builds on that earlier work in two ways. First, it substantially recasts the language and model of computation into the more standard Halpern-Tuttle framework for reasoning about knowledge and probability. Second, it brings together two distinct characterizations of security from that work. One was equivalent to the information-theoretic security criterion for a system to be free of covert channels but was difficult to prove. The other was a verification condition that implied the first; it was more easily provable but was too strong. This paper presents a characterization that is syntactically very similar to our previous verification condition but is proven to be semantically equivalent to the security criterion. The new characterization also means that our security criterion is expressible in a simpler logic and model.

1 Introduction

Multilevel security is the aspect of computer security concerned with protecting information that is classified with respect to a multilevel hierarchy (e.g., UNCLASSIFIED, SECRET, TOP SECRET). A *probabilistic system* is a hardware or software system that makes probabilistic choices (e.g., by consulting a random number generator) during its execution. Such probabilistic choices are useful in a multilevel security context for

introducing noise to reduce the rate of (or eliminate) illicit communication between processes at different classification levels. In this paper, we are concerned with definitions of *perfect* (information-theoretic) multilevel security in the sense that the definitions rule out *all* illicit communication without relying on any complexity-theoretic assumptions. That is, our model allows the system penetrators to have unlimited computational power and yet, our definitions are sufficient to ensure that there can be no illicit communication.

The motivation for reasoning about the probabilistic behavior of systems has appeared in examples and discussions of many authors (cf. [Bro91, Gra92, MR88, McC88, McL90, WJ90]). Essentially, the motivation is that it is possible for a probabilistic system to satisfy many existing definitions of security (e.g., Sutherland's *Nondeducibility* [Sut86], McCullough's *Restrictiveness* [McC90], etc.) and still contain probabilistic covert channels.

A primary contribution of our earlier work [GS92] was the unification of the logical approach to multilevel security developed by Glasgow, MacEwen, and Panangaden [GMP90] and Bieber and Cuppens [BC92] with the work on security of probabilistic systems done by McLean [McL90], Browne [Bro89], and Gray [Gra92]. In particular, we proved that the semantic interpretation of a logical formula due to Glasgow et al. is equivalent to Gray's *Probabilistic Noninterference* (which is itself equivalent to Browne's *Stochastic Non-Interference*). We also gave a verification condition (in our logic) and proved that it is equivalent to Gray's *Applied Flow Model* (which is closely related to McLean's *Flow Model*).

This paper builds on that earlier work in two fundamental ways. First, we present a new logic and corresponding semantics that is designed to substantially recast our previous work into the more standard Halpern-

*Supported by grants HKUST 608/94E from the Hong Kong Research Grants Council.

Tuttle framework for reasoning about probability and knowledge in computing systems [HT93]. Second, we give a logical characterization of security that does the job of a security definition while remaining very similar in form to the verification condition. It also gives a security definition in terms of simpler and more standard modalities than the definitions in [GS92].

The remainder of the paper is organized as follows. In §2 we set out our model of computation. In §§3 and 4, we set out the syntax and semantics of our logic. In §5 we state our definition of security and prove that it is equivalent to Probabilistic Noninterference. Finally, in §6, we give some conclusions of this work.

2 System Model

In this section, we describe our system model. This is the model by which we will (in §4) give semantics to our logic. First, we describe the general system model, which is taken from Halpern and Tuttle [HT93]. Then, we tailor the model to our needs by (in Halpern and Tuttle's terminology) choosing the "adversaries". Finally, we impose some additional structure on the model, resulting in our application-specific model.

2.1 General System Model

In this subsection we review the general system model of Halpern and Tuttle. A complete description of their model can be found in [HT93].

We have a set of agents, P_1, P_2, \dots, P_n , each with its own local state. The *global state* is an n -tuple of the local agents' states. A *run* of the system is a mapping of times to global states. We assume that time is discrete because we are dealing with security at the digital level of the system. We are not, for example, addressing security issues such as analog channels in hardware. Therefore, as in [HT93], we will assume that times are natural numbers.

The probabilities of moving among global states are represented in the model by means of labeled computation trees. The nodes of the trees represent global states. For any given node in a tree, the children of that node represent the set of global states that could possibly come next. Each arc from a node to one of its children is labeled with the probability of moving to that state. Thus, from any given node, the sum of the probabilities on its outgoing arcs must be one. As in [HT93], we also assume that the set of outgoing arcs is finite and that all arcs are labeled with *nonzero* probabilities. This final assumption can be viewed as a *convention* that if the probability of moving from state x to state y is zero, then state y is not included as a child of state x .

Certain events in a system may be regarded as *nonprobabilistic* (while still being nondeterministic). The typical example occurs when a user is to choose an input and in the analysis of the system, we do not wish to assign

a probability distribution to that choice; in such cases, we regard the choice as nonprobabilistic. All nonprobabilistic choices in the system are lumped into a single choice that is treated as being made by an "adversary" prior to the start of execution. Thus, after this choice is made, the system's execution is purely probabilistic. In Halpern and Tuttle's words, the nonprobabilistic choices have been "factored out".

In the model of computation, each possible choice by the adversary corresponds to a labeled computation tree. In other words, a system is represented as a set of computation trees, each one corresponding to a different choice by the adversary. There is no indication how the adversary's choice is made, just that it is made once and for all, prior to the start of execution.

2.2 Application-Specific System Model

In this section, we impose some additional structure on the general model described in the previous section. We fix the set of agents, fix our model and intuitions regarding communication, place some (environmental) constraints on the agents, and fix the set of choices available to the adversary.

AGENTS For our purposes we can limit the model to three agents: (1) the system under consideration, denoted Σ , (2) the covert senders (or alternatively, the high environment), denoted \mathcal{H} , and (3) the covert receivers (or alternatively, the low environment), denoted \mathcal{L} . In the remainder of the paper, we will tacitly assume that the global system is comprised of these three agents.

MODEL OF COMMUNICATION Our model of communication is similar to those of [BC92], [Gra92], and [Mil90]. We view Σ 's interface as a collection of channels on which inputs and outputs occur. Since we consider the agent \mathcal{H} (resp., \mathcal{L}) to consist of *all* processing that is done in the high (resp., low) environment, including any communication mechanism that delivers messages to Σ , we will not need to model messages in transit or, in Halpern and Tuttle's terminology, the state of the environment; rather, these components of the global state will be included as part of \mathcal{H} 's and \mathcal{L} 's state.

In many systems of interest, the timing of events is of concern. (See [Lam73] for an early description of covert communication channels that depend on timing; see [Wra92] for more recent work.) In such cases, we model the passage of time by taking the set of times (i.e., the domain of the runs) to be the ticks of some clock that is independent of the covert senders' and receivers' processing. For example, we may think of this clock as being Σ 's system clock. In this way we can properly account for covert channels that depend on time. Note that we are considering a worst-case scenario. This means that we consider the fastest way that adversaries might synchronize as a clock. If they can-

not find a common clock, they cannot communicate by means that depend on timing.

Since the mechanisms of high-level¹ I/O routines may introduce covert channels (see, e.g., [McC88, §2.3]), we take a very low-level view of I/O. In particular, we assume one input and one output per channel per unit time. That is, for each time we have a vector of inputs (one for each channel) and a vector of outputs (one for each channel). If a given agent produces no new data value at a given time, it may in fact serve as a signal in a covert channel exploitation. Hence, we treat such “no new signal” events as inputs. Similarly, we do not consider the possibility that the system can prevent an input from occurring. Rather, the system merely chooses whether to make use of the input or ignore it. Any acknowledgement that an input has been received is considered to be an output.

Given these considerations, we fix our model of communication as follows. We assume the following basic sets of symbols, all nonempty:

C : a finite set of input/output channel names,
 c_1, \dots, c_k ,

I : representing the set of input values,

O : representing the set of output values, and

\mathbb{N}^+ : representing the set of positive natural numbers.
This set will be used as our set of “times”.

Since there is one input per channel at each time, we will be talking about the vector of inputs that occurs at a given time. We will denote the set of all vectors of inputs by $I[C]$. Typical inputs vectors will be denoted $a, a', a_1, \dots \in I[C]$.

Similarly, we will denote the set of all output vectors by $O[C]$ and typical output vectors will be denoted $b, b', b_1, \dots \in O[C]$.

To talk about the history of input vectors up to a given time, we introduce notation for traces. We will denote the set of input traces of length k by $I_{C,k}$. Mathematically, $I_{C,k}$ is a shorthand for the set of functions from $C \times \{1, 2, \dots, k\}$ to I . Therefore, for a trace $\alpha \in I_{C,k}$, we will denote the single input on channel $c \in C$ at time $k' \leq k$ by $\alpha(c, k')$.

We will also need to talk about infinite traces of inputs. For this we use the analogous notation $I_{C,\infty}$, which is shorthand for the set of functions from $C \times \mathbb{N}^+$ to I .

Similarly, we will denote the set of output traces of length k by $O_{C,k}$ and the set of infinite output traces by $O_{C,\infty}$. Naturally, for an output trace β , $\beta(c, k)$ represents the output on channel c at time k .

¹In this context, “high-level” means highly *abstract* rather than highly *classified*.

There will be situations when we want to talk about vectors or traces of inputs or outputs on some subset of the channels, $S \subseteq C$. In such cases we will use the natural generalizations of the above notations, viz, $I[S]$, $I_{S,k}$, $I_{S,\infty}$, etc.

ENVIRONMENTAL CONSTRAINTS Any given agent will be able to see the inputs and outputs on a subset of the channels. We make this precise by “restricting” vectors and traces to subsets of C . Given an input vector $a \in I[C]$ and a set of channels $S \subseteq C$, we define $a \upharpoonright S \in I[S]$ to be the input vector on channels in S such that $a \upharpoonright S(c) = a(c)$ for all $c \in S$.

Similarly, given an input trace $\alpha \in I_{C,k}$ and a set of channels $S \subseteq C$, we define $\alpha \upharpoonright S \in I_{S,k}$ to be the input trace for channels in S such that $\alpha \upharpoonright S(c, k') = \alpha(c, k')$ for all $c \in S$ and all $k' \leq k$.

We assume that the set of low channels, denoted L , is a subset of C . Intuitively, L is the set of channels that the low environment, \mathcal{L} , is able to directly see. In particular, \mathcal{L} is able to see both the inputs and the outputs that occur on channels in L .

In practice, there will be some type of physical or procedural constraint on the agent \mathcal{L} to prevent it from directly viewing the inputs and outputs on channels in $C - L$. On the other hand, we place no constraints on the set of channels that \mathcal{H} is able to see. In particular, we make the worst-case assumption that \mathcal{H} is able to see all inputs and outputs on all channels. These considerations are consistent with what we’ve called the “Secure Environment Assumption” in previous work [Gra92, GS92]. In the present paper, this assumption is made precise in terms of our definition of the adversary to be given next.

THE ADVERSARY As discussed above, in Halpern and Tuttle’s framework, all nonprobabilistic choices are factored out of the execution of the system by fixing an adversary at the start of execution. To make use of this framework, we must define the set of possible adversaries from which this choice is made.

The “adversary” in our application is the pair of agents, \mathcal{H} and \mathcal{L} , that are attempting to send data from the high environment across the system Σ to the low environment. To be fully general, we model these agents as mixed strategies (in the game-theoretic sense). That is, at each point in the execution of the system the strategy gives the probability distribution over the set of next possible inputs, conditioned on the history up to the current point. In the next section, we present an example to motivate the need for such generality. Before doing that, we make the adversary precise with the following two definitions.

Definition 2.1 An *adversary* is a conditional probability function, $\mathcal{A}(a \mid \alpha, \beta, k)$. Here $a \in I[C]$ and k is some time such that there is a time k' with $k \leq k' \leq \infty$,

and $\alpha \in I_{C,k'}$ and $\beta \in O_{C,k'}$. (The k indicates that the probability of a is conditional only on the restriction of α and β to k .) \square

Intuitively, the adversary describes the environment's conditional distribution on the next input vector, given the previous history of inputs and outputs. For example, at $k = 0$, $\mathcal{A}(a \mid \alpha, \beta, k)$ gives the probability of the environment producing a at the first time unit, given the empty history.

Later in this section, we describe how a given adversary \mathcal{A} and the description of a particular system, Σ , are used to construct the corresponding computation tree $T_{\mathcal{A}}$.

Definition 2.2 We say that an adversary \mathcal{A} satisfies the *Secure Environment Assumption with respect to a set of channels* $L \subseteq C$ iff there exists a pair of conditional probability functions \mathcal{H} and \mathcal{L} such that for all $a \in I[C]$, $k \in \mathbb{N}^+$, all $\alpha \in I_{C,k}$, and all $\beta \in O_{C,k}$,

$$\mathcal{A}(a \mid \alpha, \beta, k) = \mathcal{H}(a \mid (C - L) \mid \alpha, \beta, k) \cdot \mathcal{L}(a \mid L \mid \alpha \mid L, \beta \mid L, k)$$

(where \cdot denotes real multiplication). \square

The Secure Environment Assumption can be intuitively understood as saying that the input on channels in $C - L$ at time k is (conditionally) statistically independent of the input on channels in L at time k , and the input on channels in L at time k depends only on previous inputs and outputs on channels in L . For the remainder of this paper, we will assume that all adversaries satisfy the Secure Environment Assumption.

Since there is one tree for each possible adversary, we can think of the set of trees as being indexed by the adversaries. Therefore, we will often write $T_{\mathcal{A}}$, $T_{\mathcal{A}'}$, $T_{\mathcal{A}''}$, etc.

It is clear that for an adversary \mathcal{A} that satisfies the Secure Environment Assumption (wrt L), the conditional probability functions \mathcal{H} and \mathcal{L} that must exist are in fact unique. Further, given \mathcal{H} and \mathcal{L} , there is a unique adversary, \mathcal{A} , for which \mathcal{H} and \mathcal{L} are the probability functions that satisfy the corresponding constraint. We may therefore sometimes write $T_{\mathcal{H},\mathcal{L}}$, $T_{\mathcal{H}',\mathcal{L}'}$, etc. when we want to refer to the parts of the adversary individually.

Note that our definition of an adversary is not meant to be as general as the adversary discussed by Halpern and Tuttle. (In fact, Halpern and Tuttle give no structure at all to their adversary.) Rather, our adversary is application-specific; in particular, it is for reasoning about multilevel security of probabilistic systems and is not designed to be used outside that domain.

On the other hand, this particular adversary represents a novel application of Halpern and Tuttle's framework.

In their examples, the adversary represents one or both of two possible things:

- the initial input to the system; and
- the schedule according to which certain events (e.g., processors taking steps) occur.

In contrast, our adversary does not represent a given input to the system. Rather, it represents a mixed strategy for choosing the inputs to the system. In some sense, we can think of this as a generalization on the first item above; however, our application still fits within the framework set out by Halpern and Tuttle.

THE STATE OF THE SYSTEM At any given point, P , in any given computation tree, $T_{\mathcal{A}}$, there should be a well-defined state of the system. For our purposes, the state includes the following information.

1. All inputs and outputs that have occurred on all channels up to the current time.
2. Following [HT93], we make the assumption that all points in all trees are unique by assuming that the state encodes the adversary. That is, all nodes in tree $T_{\mathcal{A}}$ encode \mathcal{A} . Note that we do *not* assume that any given agent knows the adversary; just that it is somehow encoded in the state. We can think of the high part of the adversary, \mathcal{H} , as being encoded in the high environment and the low part, \mathcal{L} , as being encoded in the low environment.
3. Typically, there are additional components of the global state representing the internal state of Σ . For example, in describing Σ , it is often convenient to use internal state variables. The state of these variables can be thought of as a vector of values, one value for each state variable. The internal state, when it exists, will be denoted c , and the history of internal states will be denoted γ .

COMPUTATION TREES Now that we have set out the possible states of the system (i.e., the points of computations), we can talk about the construction of the computation trees.

For each reachable point, P , we assume that Σ 's probability distribution on outputs is given. For example, this can be given by a conditional probability distribution, $\mathcal{O}(b, c \mid \alpha, \beta, \gamma, k)$, where c is the vector representing values of all internal state variables (i.e., the internal system state) at time $k + 1$, $b \in O[C]$ is the vector of outputs produced by the system at $k + 1$, and α, β, γ give the history through k of inputs, outputs, and internal state values, respectively.

Given $\mathcal{O}(b, c \mid \alpha, \beta, \gamma, k)$ and the adversary, \mathcal{A} we can construct the corresponding computation tree by starting with the initial state of the system (i.e., the point at

the root of the tree with empty histories of inputs, outputs, etc.) and iteratively extending points as follows.

Let P be a point in the tree with internal system history γ , input history α , and output history β . We will make P' a child of P iff

1. P' is formed from P by modifying the internal system state to c and extending P 's input history (output history, resp.) with a (b , resp.); and
2. both $\mathcal{O}(b, c \mid \alpha, \beta, \gamma, k)$ and $\mathcal{A}(a \mid \alpha, \beta, k)$ are positive.

In such cases, we label the arc from P to P' with $\mathcal{O}(b, c \mid \alpha, \beta, \gamma, k) \cdot \mathcal{A}(a \mid \alpha, \beta, k)$, i.e., the system, Σ , and the environment, \mathcal{A} , make their choices independently.

RUNS OF THE SYSTEM A run of the system is an infinite sequence of states along a path in one of the computation trees. When we want to talk about the particular run, ρ , and time, k , at which a point P occurs, we will denote the point by the pair (ρ, k) . Further, if we wish to talk about the various components of the run, i.e., the trace of the inputs, α , outputs, β , or other variables, γ , we will denote the run by (α, β, γ) and denote the point, P , by $(\alpha, \beta, \gamma, k)$.

For a given tree, T , we denote the set of runs (i.e., infinite sequences of states), formed by tracing paths from the root, by $runs(T)$.

For security applications we are concerned with information flow into and out of the system rather than with information in the system per se. Thus, though our system model is adequate to represent internal states and traces thereof, in subsequent sections it will be adequate to represent systems entirely in terms of input and output. For example, system behavior at time k can be represented by ' $\mathcal{O}(b \mid \alpha, \beta, k)$ ' rather than ' $\mathcal{O}(b, c \mid \alpha, \beta, \gamma, k)$ '.

3 Syntax

In this section we set out our formal language and use it to describe two simple systems. Then we give the axioms and rules of our logic.

3.1 Formation Rules

To describe the operation of the system under consideration (viz, Σ), we use a variant of Lamport's Raw Temporal Logic of Actions (RTL) [Lam91].² The primary difference is that we add a modal operator $Pr_i(\varphi)$ that allows us to specify and reason about the probabilistic behavior of the system.

²Roughly speaking, Raw Temporal Logic of Actions (RTL) is the same as Lamport's Temporal Logic of Actions (TLA) without the treatment of stuttering [Lam91]. Since we are not, in this paper, concerned with refinement, we omit the considerations of stuttering and use RTL.

From the previous section, we assume the following basic sets of symbols, all nonempty: C , I , O , and also \mathbb{R} . Members of \mathbb{R} will have the usual representation—e.g., $43.5 \in \mathbb{R}$.

We will also be talking about the subjects (or agents) of the system. Formally, a *subject*, $S \subseteq C$, is identified with the process's view of the system, i.e. the set of channels on which it can *see* the inputs and outputs.

Formulae in the language are built up according to the following rules.

- constants from the set of basic symbols are terms.
- state variables (representing the value of that variable in the current state) are terms. Among the state variables, there are two reserved for each communication channel. For each $c \in C$, we have a state variable c_{in} that takes values from I , and another state variable c_{out} that takes values from O . Note that, implicitly, *inputs* are from the covert senders and receivers into the system (Σ) and *outputs* are from the system to the covert senders and receivers. This is because Σ is the system under consideration (i.e., with respect to which we are reasoning about security). We have no mechanism (and no need) to specify communication between agents not including the system under consideration.
- primed state variables (e.g., c'_{in}) are terms. (These represent the value of the variable in the next state.)
- We use standard operators among terms (e.g., $+$ and \cdot for addition and multiplication, respectively), with parentheses for grouping subterms, to form composite terms.
- an atomic *predicate* is an equation or inequality among terms *not* containing primed state variables.
- an atomic *action* is an equation or inequality among terms (possibly including primed as well as unprimed state variables). (Note that all predicates are actions.)
- for any action, φ , and for any subject $S \subseteq C$, $Pr_S(\varphi)$ is a real-valued term (representing the subjective *probability* that S assigns to the formula φ).
- For any predicate, φ , φ is a temporal formula.
- For any action or temporal formula φ , $\Box\varphi$ is a temporal formula (to be read intuitively as *always* φ).
- We build up composite predicates, actions, and temporal formulae, resp., in the usual recursive fashion using \wedge , \vee , \neg , and \rightarrow .

Now, to specify and reason about our security properties of interest, we add three finite sets of modal operators on formulae: K_1, \dots, K_n , $\mathbf{K}_1, \dots, \mathbf{K}_n$, and R_1, \dots, R_n , representing knowledge of a (relatively) weak subject, knowledge of a powerful subject, and permitted-knowledge respectively for each subject (represented by the subscript of the operator). Therefore, we add the following formation rules to our syntax.

- For any action (temporal formula, resp.) φ , and for any subject $S \subseteq C$, $K_S(\varphi)$ (representing that the weak subject S knows φ), $\mathbf{K}_S(\varphi)$ (representing that the powerful subject S knows φ) and $R_S(\varphi)$ (representing that S has *permitted* knowledge of φ) are actions (temporal formulae, resp.).

Later in the paper, we will make the meaning of these three operators precise. For now, we merely mention that the weak-subject knowledge operators (K_S) will be given the standard semantics (e.g., as in [HT93]); the powerful-subject knowledge operators (\mathbf{K}_S) will be given semantics that imply greater knowledge on the part of the subject (viz, knowledge of the probability of certain future events).

3.2 Examples

We now give two simple examples of how to describe systems in our language. Ultimately, we will have sufficient formal machinery to show that one of these systems is secure and the other is not; however, here we simply set them out formally. These descriptions are meant to give the reader an intuitive feel for the meaning of expressions in the language. Precise meanings will be given in §4. Also, the second of these examples will motivate our choice of modeling adversaries as strategies.

Example 3.1 The first example is a simple encryption box that uses a “one-time pad” [Den82]. It has two channels, *high* and *low*. At each tick of the system clock, it inputs a 0 or 1 on the high channel and outputs a 0 or 1 on the low channel. The low output is computed by taking the “exclusive or” (denoted \oplus) of the high input and a randomly generated bit. It is well known that this results in an output stream that is uniformly distributed. Therefore, we can describe this system as follows.

Let $C = \{h, l\}$, $I = \{0, 1\}$, and $O = \{0, 1\}$. Then, the system is specified by the following formula.

$$\Box (Pr_C(l'_{out} = 0) = Pr_C(l'_{out} = 1) = 0.5)$$

In this formula, l_{out} is a state variable representing the output on the low channel, l . Therefore, l'_{out} is the output on l at the *next* time. Further, $Pr_C(l'_{out} = 0)$ denotes the probability that the output on l is a 0 at the next time. Hence, the entire formula says that at all

times, the probability of Σ producing a one (1) on the next clock tick is equal to the probability of producing a zero (0), which is equal to 0.5. Note that we have not specified anything about the probability distribution on inputs, since that is part of the environment behavior rather than the system behavior. \square

Example 3.2 The second example is an insecure version of the simple encryption box. This system was first described by Shannon in [Sha58].

As in the first example, at each tick, Σ computes the “exclusive or” of the high input and a randomly generated bit and sends that value out on the low channel. However, in this system, the randomly generated bit used at any given tick is generated and sent out on the high output channel during the *previous* tick of the clock.

This can be expressed in our formalism as follows. Let $C = \{h, l\}$, $I = \{0, 1\}$, and $O = \{0, 1\}$. The following formula specifies the system.

$$\Box ((Pr_C(h'_{out} = 0) = Pr_C(h'_{out} = 1) = 0.5) \wedge (l'_{out} = h_{out} \oplus h'_{in}))$$

Note that in the second conjunct, h_{out} is unprimed, indicating that the output on l at the *next* time is the “exclusive or” of the *current* output on h with the *next* input on h .

Now note that if the high agent ignores its output, then this system acts exactly as the system from the previous example (and can be used for perfect encryption). In particular, suppose we were to model an adversary as an input string—the input to be provided by the high agent. Then, it is easy to see that for any adversary (i.e., any high input string) fixed prior to the start of execution, the output to low will be uniformly distributed and, in fact, will contain no information about the high input string.

However, the bit that will be used as the one-time pad at time t is available to the high agent at time $t - 1$. Therefore, (due to the algebraic properties of “exclusive or”, viz, $x \oplus x \oplus y = y$) the high agent can use this information to counteract the encryption. In particular, the high agent can employ a (game-theoretic) strategy to send any information it desires across the system to the low agent.

For example, suppose the high agent wishes to send a sequence of bits, b_1, b_2, \dots . We’ll denote the high input (resp., output) at time k by $h_{in}(k)$ (resp., $h_{out}(k)$). The appropriate strategy for the high agent is as follows.

The high agent chooses its input for time $k + 1$ as $h_{in}(k + 1) = h_{out}(k) \oplus b_k$.

Thus, the output to low at time $k + 1$, denoted $l_{out}(k + 1)$ is computed as follows.

$$\begin{aligned} l_{out}(k+1) &= h_{out}(k) \oplus h_{in}(k+1) \\ &= h_{out}(k) \oplus h_{out}(k) \oplus b_k \\ &= b_k \end{aligned}$$

The first line follows from the system description, the second from the high strategy, and the third from the properties of \oplus . Thus, by employing the correct strategy, the high agent can noiselessly transmit an arbitrary message over Σ to the low agent. This, of course, motivates our choice of strategies as the adversary, rather than, e.g., input strings.

□

We now have some sense of the formal language, with the exception of the modal operators K_S , \mathbf{K}_S , and R_S . As previously mentioned, these operators are used to formalize the security properties that interest us; so, we will discuss their use in a later section. First, we describe the logical axioms and inference rules that are used to prove properties about systems.

3.3 The Logic

We now give the axioms of our logic. In the following, we will use ' φ ' and ' ψ ' to refer to formulae of our language.

Propositional Reasoning All instances of tautologies of propositional logic.

Temporal Reasoning The following are standard axioms for temporal reasoning about discrete systems. The logic they constitute is generally called *S4.3Dum* or sometimes **D**. (See [Gol92] for details. Note also that these are the formulae Abadi uses to axiomatize Lamport's TLA [Aba90].) We have labeled the axioms with their historical names. Let φ and ψ be formulae of our language.

$$\begin{aligned} \mathbf{K} \quad & \Box(\varphi \rightarrow \psi) \rightarrow (\Box\varphi \rightarrow \Box\psi) \\ \mathbf{4} \quad & \Box\varphi \rightarrow \Box\Box\varphi \\ \mathbf{D} \quad & \Box\varphi \rightarrow \Diamond\varphi \\ \mathbf{L} \quad & \Box(\varphi \wedge \Box\varphi \rightarrow \psi) \vee \Box(\psi \wedge \Box\psi \rightarrow \varphi) \\ \mathbf{Z} \quad & \Box(\Box\varphi \rightarrow \varphi) \rightarrow (\Diamond\Box\varphi \rightarrow \Box\varphi) \end{aligned}$$

' $\Diamond\varphi$ ' can be interpreted roughly as saying that at some point φ is true. Formally, it is viewed as notational shorthand: for all formulae φ , $\Diamond\varphi \triangleq \neg\Box\neg\varphi$. **K** basically guarantees that the temporal operator respects modus ponens. Each of the other axioms captures a feature of time that we desire. **4** gets us transitivity. **D** guarantees that we don't run out of time points (seriality). **L** guarantees that all points in time are connected. And, **Z** guarantees that time is discrete. (Between any two points in time there are at most finitely many other points.)

Real Number Axioms Standard field and order axioms for the real numbers (to apply to members of \mathbb{R} and function terms with range \mathbb{R} .) We will not enumerate these axioms. (See any elementary real analysis book for enumeration, e.g., [Mar74] or [Rud].)

Epistemic Reasoning The (nonredundant) axioms of the Lewis system **S5**. (cf. [Che80] or [Gol92]) apply to the strong knowledge operators (\mathbf{K}_i), the weak knowledge operators (\bar{K}_i), and the permitted-knowledge operators (R_i). We state them only for the (strong) knowledge operators. As for temporal axioms, we give the axioms their historical names. Let S be a subject, and let φ and ψ be formulae of our language.

$$\begin{aligned} \mathbf{K} \quad & [\mathbf{K}_S(\varphi) \wedge \mathbf{K}_S(\varphi \rightarrow \psi)] \rightarrow \mathbf{K}_S(\psi) \quad (\text{Knowledge respects modus ponens.}) \\ \mathbf{T} \quad & \mathbf{K}_S(\varphi) \rightarrow \varphi \quad (\text{What one knows is true.}) \\ \mathbf{5} \quad & \neg\mathbf{K}_S(\varphi) \rightarrow \mathbf{K}_S\neg\mathbf{K}_S(\varphi) \quad (\text{If you don't know something, then you know that you don't know it.}) \end{aligned}$$

We also have two axioms for relating weak knowledge to permitted knowledge and permitted knowledge to strong knowledge.

$$\begin{aligned} \mathbf{kR} \quad & K_S(\varphi) \rightarrow R_S(\varphi) \\ \mathbf{RK} \quad & R_S(\varphi) \rightarrow \mathbf{K}_S(\varphi) \end{aligned}$$

Random Variable Axioms The standard requirements for random variables (in the probability-theoretic sense).

PM (Positive Measure) for any formula, φ , and any subject, S , $Pr_S(\varphi) \geq 0$ (The probability of any event is greater than or equal to zero.)

NM (Normalized Measure) for any channel, c , and any subject, S ,

$$\begin{aligned} \sum_{i \in I} Pr_S(c_{in} = i) &= 1 \quad (\text{The probability of all possibilities sums to one.}) \\ \sum_{o \in O} Pr_S(c_{out} = o) &= 1 \end{aligned}$$

Input/Output Axioms for powerful-subject-knowledge and permitted-knowledge of inputs and outputs. Let S be a subject, let $c \in S$ be a channel that is visible to S , and let $a \in I$ be an input, $b \in O$ be an output, and $r \in \mathbb{R}$ be a real number.

$$\begin{aligned} \mathbf{KO} \quad & Pr_S(c'_{out} = o) = r \rightarrow \mathbf{K}_S(Pr_S(c'_{out} = o) = r) \\ \mathbf{RI} \quad & Pr_S(c'_{in} = i) = r \rightarrow R_S(Pr_S(c'_{in} = i) = r) \end{aligned}$$

Intuitively, **KO** say that powerful subjects know the distribution on their own outputs conditioned on the previous history of inputs and outputs they have seen.

RI says that all subjects are permitted to know the conditional distribution on their own inputs.

Note that a powerful subject knows the distribution on its own inputs conditioned on the previous history of inputs and outputs it has seen; this follows trivially from **RI** and **RK**, thus we have a theorem **KI**, which is analogous to **RI**. From theorem **KI** and axiom **KO** we can inductively show that powerful subjects know the probability of all events they can see in finite time.

On the other hand, a subject is permitted to know the conditional distribution on its own outputs only if the system is secure—e.g., for a low subject, only if knowing that distribution does not reveal any information about the distribution on high inputs. The absence of an axiom **RO**, corresponding to **KO**, is what syntactically captures this.

The above are all of our axioms. We now give the rules of our logic, which are all standard.

MP (Modus Ponens) From φ and $\varphi \rightarrow \psi$ infer ψ .

Nec (Necessitation) This rule applies to all of the modal operators we have introduced: \Box , \mathbf{K}_S , K_S , and R_S . (It is called ‘necessitation’ because it was originally applied to a necessity operator.) We set it out for \Box only. From $\vdash \varphi$ infer $\vdash \Box \varphi$.

Note that in the above, ‘ $\vdash \varphi$ ’ indicates a derivation of φ from the axioms alone, rather than from a set of premises. (Derivations will be formally defined below.) Thus, in the case of knowledge (strong or weak) for example, **Nec** says that if φ is a theorem (derivable without any premises) then all subjects know φ .

We now have sufficient machinery to give a characterization of a formal derivation.

Definition 3.3 Let Σ be a finite set of formulae of our language. A finite sequence of formulae $\varphi_1, \varphi_2, \varphi_3, \dots, \varphi_n$ is called a *derivation* (of φ_n from Σ) iff each φ_k ($k = 1, \dots, n$) satisfies one of the following:

- $\varphi_k \in \Sigma$,
- φ_k is an axiom.
- φ_k follows from some theorem by **Nec**.
- For some $i, j < k$, φ_k results from φ_i and φ_j by **MP**.

We write ‘ $\vdash \varphi$ ’ to indicate a derivation of φ from Σ , and we write ‘ $\vdash \varphi$ ’ to indicate a derivation of φ from the axioms alone. \square

This completes our statement of the formal system.

4 Semantics

In the last section we presented a syntactic system. So far we have only intuitive meanings to attach to this formalism. In this section we provide semantics for our system in terms of the Halpern-Tuttle framework and our application-specific model set out in §2.

4.1 Semantic Model

A *model* M is a tuple of the form:

$$\langle \mathbb{R}, +, \cdot, \leq, W, T, C, I, O, v, \kappa_1^{powerful}, \dots, \kappa_{|\mathcal{P}(C)|}^{powerful}, \kappa_1^{weak}, \dots, \kappa_{|\mathcal{P}(C)|}^{weak}, \delta_1, \dots, \delta_{|\mathcal{P}(C)|} \rangle$$

Here, \mathbb{R} and its operations and ordering relation gives us the real numbers; W is the set of worlds (i.e., global states); T is the set of labeled computation trees (with nodes from W); C , I , and O are the sets of channels, possible inputs, and possible outputs, respectively; v is the assignment function, which assigns semantic values to syntactic expressions at each world; (values of v at a particular world P , will be indicated by the projection ‘ v_P ’); the $\kappa_{i_S}^{powerful}$ and $\kappa_{i_S}^{weak}$ are knowledge accessibility relations, one each for each subject S ; and the δ_{i_S} are permitted-knowledge accessibility relations, also one for each subject. In the remainder of this paper we will generally denote the accessibility relations corresponding to subject S by ‘ $\kappa_S^{powerful}$ ’, ‘ κ_S^{weak} ’, and δ_S ’. These will each be further explained when we come to the assignment function.

In assigning meaning to our language, it is of fundamental importance to associate a probability space with each labeled computation tree. In particular, for each labeled computation tree T_A we will construct a sample space of runs, \mathcal{R}_A , an event space, \mathcal{X}_A (i.e., those subsets of \mathcal{R}_A to which a probability can be assigned) and a probability measure μ_A that assigns probabilities to members of \mathcal{X}_A .

Our construction of this probability space is quite natural and standard (see, e.g., [Sei92] as well as [HT93] for two instances). We will not go into detail explaining the basic concepts of probability and measure theory here (cf. [Hal50] or [Shi84]).

Definition 4.1 For a labeled computation tree T_A , the associated **sample space** \mathcal{R}_A is the set of all infinite paths starting from the root of T_A .

The set $e \subseteq \mathcal{R}_A$, is called a **generator** iff it consists of the set of all traces with some common finite prefix. The generators are the probability-theoretic events corresponding to finite traces. We can now define the **event space**, \mathcal{X}_A , to be the (unique) field of sets generated by the set of all generators (i.e., \mathcal{X}_A is the smallest subset of $\mathcal{P}(\mathcal{R}_A)$ ³ that contains all of the generators and is closed under countable union and complementation).

³ \mathcal{P} denotes “powerset”.

Suppose e is a generator corresponding to the finite prefix given by (ρ, k) . Then, the probability measure, μ_A , is defined for e as the product of the transition probabilities from the root of the tree, along the path ρ , up to time k . Further, there is a unique extension of μ_A to the entire event space [Hal50]. \square

4.2 Assignment Function

For a given point, P , we will assign truth values to temporal formulae φ at this point. In addition, we assign values to variables, for example the input on a channel, at this point. The assignment function that does both of these is denoted by v_P .

To define v_P , we will need to assign truth values to action and temporal formulae. Therefore we will also define functions $v_{(P_1, P_2)}$ (where P_1 and P_2 are points) and v_ρ (where ρ is a run) to assign truth values to action formulae over a pair of points and temporal formulae on a run, respectively.

We define v_P , $v_{(P_1, P_2)}$, and v_ρ mutually recursively below. First we present some additional notation.

Notation Since nodes are unique even across trees, for a given node P , there is no ambiguity in referring to “the tree that contains P ”. In the following, we will use $tree(P)$ to denote that tree.

Further, since there is a one-to-one correspondence from trees to adversaries, we can refer to “the adversary corresponding to $tree(P)$ ”. We denote that adversary by $A(P)$.

We use the notation $succ(P)$ to denote the set of nodes that succeed P in $tree(P)$.

We use the notation $extensions(P)$ to denote the set of infinite sequences of states starting at P in $tree(P)$.

As discussed in [HT93], to each subject, S , and point, P , we need to associate a sample space, $\mathcal{S}_{S,P}$. Each such sample space will be a set of points from $tree(P)$. Intuitively, these are the points (within the tree that contains the current execution) that the subject S considers possible. We will set out these sample spaces below. For the time being, we simply make use of the notation $\mathcal{S}_{S,P}$ to refer to them.

We will be rather abusive in the use of our probability measures μ_A . In particular, when we have a finite set of points, x , we will write $\mu_A(x)$ to denote the probability (as assigned by μ_A) of passing through one of the points in x . Technically, this is wrong, since μ_A is defined for (certain) sets of runs; not for sets of points. However, the mapping between the two is extremely natural; the set of runs corresponding to a point is the set of runs that *pass through* that point. Further, by the construction of our probability spaces, all sets of runs corresponding to finite sets of points are measurable. Therefore, there is no danger in this abuse of notation and it greatly simplifies our presentation.

As is standard (see, e.g., [HT93]), we will be using *accessibility relations*—one for each subject—on points to give semantics to our three knowledge operators. We define these relations below. For the time being, we simply make use of the notation $\kappa_S^{powerful}$ to refer to the powerful-subject knowledge accessibility relation, κ_S^{weak} to refer to the weak-subject knowledge accessibility relation, and δ_S to refer to the permitted-knowledge accessibility relation. \square

We now define v_P , $v_{(P_1, P_2)}$, and v_ρ . Let P be a point at time k in the execution $\rho = (\alpha, \beta, \gamma)$ in computation tree T_A .

- Numbers are assigned to number names.
- Members of I and O are assigned to their syntactic identifiers.
- For any channel $c \in C$,

$$v_P(c_{in}) \triangleq \alpha(c, k)$$

- For any channel $c \in C$,

$$v_P(c_{out}) \triangleq \beta(c, k)$$

- For any variable name, X , excluding channel variables (such as c_{in} or c_{out})

$$v_P(X) \triangleq \gamma(X, k)$$

- To assign truth values to actions, we need to assign values to terms at *pairs of points*. Constants do not change their values when we move to pairs of points. However, primed and unprimed variables are evaluated differently. For any state variable, X , and any pair of points (P_1, P_2) ,

$$v_{(P_1, P_2)}(X) \triangleq v_{P_1}(X)$$

In contrast,

$$v_{(P_1, P_2)}(X') \triangleq v_{P_2}(X)$$

$$v_{(P_1, P_2)}(\varphi) \triangleq v_{P_1}|_{P_2}(\varphi)$$

where $v_{P_1}|_{P_2}(\varphi)$ follows v_{P_1} except that all primed terms are assigned according to v_{P_2} .

- Composite terms are assigned values at a point and at a pair of points in the natural way. For example,

$$v_P(X + Y) \triangleq v_P(X) + v_P(Y)$$

and

$$v_{(P_1, P_2)}(X + Y) \triangleq v_{(P_1, P_2)}(X) + v_{(P_1, P_2)}(Y)$$

- Similarly, predicates and action formulae are assigned truth values at a point and at a pair of points, respectively, in the natural way. For example,

$$v_P(X \leq Y) = \mathbf{true} \text{ iff } v_P(X) \leq v_P(Y)$$

and

$$v_{(P_1, P_2)}(\varphi \wedge \psi) = \mathbf{true} \text{ iff } v_{(P_1, P_2)}(\varphi) = \mathbf{true} \text{ and } v_{(P_1, P_2)}(\psi) = \mathbf{true}$$

- An action formula, φ , is true at a point, P , iff it is true for all pairs of points emanating from P . More precisely,

$$v_P(\varphi) = \mathbf{true} \text{ iff } \forall P' \in \text{succ}(P), v_{(P, P')}(\varphi) = \mathbf{true}$$

(Since we have not needed to include quantification in our language we are free to use ‘ \forall ’ and ‘ \exists ’ as metalinguistic shorthand.)

- To interpret the *probability* of an action φ at a point P , we will take the set of all pairs of points, (P_1, P_2) emanating from points in $\mathcal{S}_{S, P}$. Restricting to this set, we compute the probability of those pairs such that $v_{(P_1, P_2)}(\varphi)$ evaluates to true. More precisely, for any action formula, φ , and for any subject $S \subseteq C$,

$$v_P(\text{Pr}_S(\varphi)) \triangleq \mu_{\mathcal{A}(P)}(\mathcal{S}_{S, P}(\varphi) \mid \mathcal{S}_{S, P})$$

where

$$\mathcal{S}_{S, P}(\varphi) \triangleq \{P_2 \mid \exists P_1 \in \mathcal{S}_{S, P} \text{ such that } P_2 \in \text{succ}(P_1) \text{ and } v_{(P_1, P_2)}(\varphi) = \mathbf{true}\}.$$

- For any predicate, φ , and run, ρ ,

$$v_\rho(\varphi) \triangleq v_{(\rho, 1)}(\varphi)$$

- For any (action or temporal) formula, φ , and run, ρ ,

$$v_\rho(\Box \varphi) = \mathbf{true} \text{ iff } \forall i, v_{(\rho, i)}(\varphi) = \mathbf{true}$$

- A temporal formula is true at a point iff it is true in all runs extending from that point. More precisely, for any temporal formula, φ ,

$$v_P(\varphi) \triangleq \forall \rho \in \text{extensions}(P), v_\rho(\varphi)$$

- Composite action formulae and temporal formulae are assigned truth values at points in the natural way. For example,

$$v_P(\varphi \wedge \psi) = \mathbf{true} \text{ iff } v_P(\varphi) = \mathbf{true} \text{ and } v_P(\psi) = \mathbf{true}$$

- Our three knowledge operators are all S5 modal operators and are given semantics in terms of the accessibility relations (on points) in the standard way; viz, for powerful-subject knowledge,

$$v_P(\mathbf{K}_S(\varphi)) = \mathbf{true} \text{ iff } \forall P', \kappa_S^{\text{powerful}}(P, P') \Rightarrow v_{P'}(\varphi) = \mathbf{true}$$

for weak-subject knowledge,

$$v_P(K_S(\varphi)) = \mathbf{true} \text{ iff } \forall P', \kappa_S^{\text{weak}}(P, P') \Rightarrow v_{P'}(\varphi) = \mathbf{true}$$

and for permitted knowledge,

$$v_P(R_S(\varphi)) = \mathbf{true} \text{ iff } \forall P', \delta_S(P, P') \Rightarrow v_{P'}(\varphi) = \mathbf{true}$$

To complete our semantics for probability formulae, we need to choose the sample spaces $\mathcal{S}_{S, P}$ for each subject at each point. Our approach is quite straightforward. We will choose $\mathcal{S}_{S, P}$ to be the set of points within $\text{tree}(P)$ that have the same history of inputs and outputs on channels S as occur on the path to point P . More precisely, we have the following definitions.

Definition 4.2 Let $S \subseteq C$ be a subject and let $\rho_1 = (\alpha_1, \beta_1, \gamma_1)$ and $\rho_2 = (\alpha_2, \beta_2, \gamma_2)$ be two runs (not necessarily in the same tree). We say that ρ_1 and ρ_2 have the same S -history up to time k if and only if $\forall i, 1 \leq i \leq k, \forall c \in S,$

$$\alpha_1(c, i) = \alpha_2(c, i) \text{ and } \beta_1(c, i) = \beta_2(c, i)$$

□

Definition 4.3 Let $S \subseteq C$ be a subject and let $P_1 = (\rho_1, k_1)$ and $P_2 = (\rho_2, k_2)$ be two points (not necessarily in the same tree). We say that P_1 and P_2 have the same S -history if and only if the following two conditions hold.

1. $k_1 = k_2$.
2. ρ_1 and ρ_2 have the same S -history up to time k_1 .

□

Definition 4.4 Let $S \subseteq C$ be a subject and P be a point; the *sample space for S at point P* is given by

$$\mathcal{S}_{S, P} \triangleq \{ P' \mid \text{tree}(P') = \text{tree}(P) \text{ and } P' \text{ and } P \text{ have the same } S\text{-history} \}$$

□

In a more general setting, we would also want to consider the possibility that a subject S has internal state variables and could use these to make finer distinctions between points. However, in our application, all of the internal processing of the relevant subjects (viz, \mathcal{H} and \mathcal{L}) is encoded in the adversary and is thus factored out

of the computation tree. We therefore do not lose any needed generality in making the above definition.

Now, to complete our description of the assignment function we need only describe the relations $\kappa_S^{powerful}$, κ_S^{weak} , and δ_S for all $S \subseteq C$.

Definition 4.5 Our definition of κ_S^{weak} (and hence our definition of weak-subject knowledge) is the standard definition of knowledge in a distributed system. In particular, for any two points, P_1 and P_2 (not necessarily in distinct trees) and any subject, $S \subseteq C$, We say that P_2 is *weak-subject-accessible* from P_1 , denoted ' $\kappa_S^{weak}(P_1, P_2)$ ', if and only if P_1 and P_2 have the same S -history. \square

Our definition of $\kappa_S^{powerful}$ (and hence, our definition of powerful-subject knowledge) is novel. In the analysis of distributed protocols and in other areas of computer science, it is typical to use the above weak-subject knowledge accessibility relation (or something roughly equivalent). Our definition of accessibility for *powerful-subject* knowledge will require more—in other words, using this definition subjects *know* more. In particular, subjects “know” the probability distribution over the future inputs and outputs on the channels that they can see. That is, if the probability of a given future output on a low channel is x , then (assuming a powerful subject) the low environment knows that. To make this notion precise, we need some definitions.

Definition 4.6 Let $S \subseteq C$ be a subject and let e be a set of runs, $\{\rho_i\}$, (not necessarily taken from any one computation tree). We say that e is an S -event if and only if there exists a time $k \in \mathbb{N}^+$ such that for any two runs, ρ_1 and ρ_2 , having the same S -history up to time k , $\rho_1 \in e$ iff $\rho_2 \in e$.

For an S -event, e , we will refer to the least k such that above condition holds as the *length* of e . \square

Intuitively, an event e is an S -event if and only if there is some finite time k (i.e., its length) after which S can always determine whether or not e has occurred.

Note that in general, an S -event contains runs from more than one computation tree. Therefore, such “events” will not be measurable in any of our probability spaces. Rather, we think of them as *meta events* and we will be interested in the measure of the subset of the runs that are contained in a given computation tree. To make this precise, we introduce the following definition.

Definition 4.7 Given a computation tree, T_A , and an S -event, e , the *projection of e onto T_A* , denoted e_A , is given by:

$$e_A \triangleq \text{runs}(T_A) \cap e$$

\square

When it is clear from context what is meant, we may occasionally confuse the meta-event with its projection, e.g., we might write ' $\mu_A(e)$ ' for ' $\mu_A(e_A)$ '.

Observation 4.8 Every projection of every S -event is measurable. That is, for any S -event, e , and any computation tree, T_A ,

$$e_A \in \mathcal{X}_A$$

This is due to the restriction on S -events that they be observable within some finite time. In particular, the projection of an S -event onto a tree, T , must also be observable within a finite time, and so it must be formable from a finite number of unions and complementations of the generators of T . \square

Now we are ready to give the definition of the knowledge accessibility relation.

Definition 4.9 Let P_1 and P_2 be two points in (not necessarily distinct) trees T_{A_1} and T_{A_2} , respectively and let $S \subseteq C$ be a subject. We say that P_2 is *powerful-subject-accessible* from P_1 , denoted ' $\kappa_S^{powerful}(P_1, P_2)$ ', iff

1. P_1 and P_2 have the same S -history; and
2. for any S -event e , $\mu_{A_1}(e|\mathcal{S}_{S,P_1}) = \mu_{A_2}(e|\mathcal{S}_{S,P_2})$

\square

Thus, when two points are $\kappa_S^{powerful}$ -accessible, this implies not only that the two points have the same S -history, but also, conditioned on the current S -history, the probability distribution on all S -events, including future events, is the same. As mentioned previously, using this definition, subjects “know more” than when using the standard definition. However, we view this as another case where we’ve adopted the worst-case scenario; that is, we’ve given the penetrators, \mathcal{H} and \mathcal{L} , the greatest conceivable knowledge at any given point in the execution of the system. We will see later in the paper that this choice corresponds to some existing information-theoretic definitions of perfect multi-level security.

Our definition of permitted knowledge is also novel. From our viewpoint, a subject’s permitted knowledge does not change over the course of the system’s execution. That is, a given subject’s permitted knowledge is set *prior* to the start of execution. (It is only a subject’s *knowledge* that changes during the system’s execution.) Thus, we can capture a subject’s permitted knowledge by defining an accessibility relation on computation trees. We will say that two *points* are accessible if and only if they have the same S -history and their two containing trees are accessible; roughly speaking, two computation trees, T_{A_1} and T_{A_2} , will be accessible if and only if the parts of the adversaries, \mathcal{A}_1 and \mathcal{A}_2 , that correspond to S “act the same” in both trees. We make this precise as follows.

Definition 4.10 Let S be a subject and $T_{\mathcal{A}_1}$ and $T_{\mathcal{A}_2}$ be two computation trees. We say that $T_{\mathcal{A}_2}$ is Δ_S -accessible from $T_{\mathcal{A}_1}$, denoted ' $\Delta_S(T_{\mathcal{A}_1}, T_{\mathcal{A}_2})$ ' iff for any point P_1 in $T_{\mathcal{A}_1}$ there is a point P_2 in $T_{\mathcal{A}_2}$ such that

1. P_1 and P_2 have the same S -history; and
2. for any channel $c \in S$ and any input $i \in I$,
 $v_{P_1}(Pr_S(c'_{in} = i)) = v_{P_2}(Pr_S(c'_{in} = i))$.

□

Definition 4.11 Let S be a subject and P_1 and P_2 be two points. We say that P_2 is δ_S -accessible from P_1 , denoted ' $\delta_S(P_1, P_2)$ ' if and only if

1. P_1 and P_2 have the same S -history; and
2. $\Delta_S(\text{tree}(P_1), \text{tree}(P_2))$.

□

Thus, the δ_S relation reflects the fact that subjects are permitted to know the conditional probability distribution on their inputs: two points are δ_S -accessible (i.e., as far as S is permitted to know they are the same point) if and only if the conditional distribution on inputs visible to S is the same at both points.

The definition of permitted knowledge and the Secure Environment Assumption combine to isolate the question that interests us: “Can the low environment (\mathcal{L}) come to know, via the system of interest (Σ), something about the activity of the high environment (\mathcal{H})?” To see how this question is captured, consider a subset L of the interface of Σ . By our definition of permitted knowledge, the low environment, \mathcal{L} , is permitted to know how the inputs on L are chosen, but *not* how other (high) inputs are chosen. Further, by the Secure Environment Assumption, \mathcal{L} cannot get any information about how high inputs are chosen via any means outside of Σ . Thus, if the low environment is able to gain some information that it is not permitted to know, it *must* have been information about the high environment and it *must* have been gained via Σ .

In the remainder of the paper, for a point P , formula φ , and set of formulae γ , we will use ' $P \models \varphi$ ' to indicate that φ is true at P , and ' $P \models \gamma$ ' to indicate that all members of γ are true at P . Finally, we will use ' $\gamma \models \varphi$ ' to indicate that φ is true at all worlds at which all members of γ are true.

4.3 Soundness

In §5 below we give a syntactic characterization of security and show that the semantic interpretation of our syntactic characterization of security is equivalent to certain previously developed information-theoretic

characterizations. However, the significance of these results is reduced unless the logic is sound. For, without soundness there is no guarantee that any formal proof of security we might give for a system implies any independently motivated notion of security. A soundness theorem gives us just such a correspondence. The above given logic is sound with respect to the above given semantics. A proof is set out in [GS95].

This completes our discussion of the logic itself. In the remainder of the paper we focus on security and applications of the logic thereto.

5 Formal Definition of Security

In this section, we give a definition of security—which we call the *Formal Security Condition* (FSC)—using the time and knowledge operators of our logic.

Definition 5.1 Let $L \subseteq C$ be a subject. Suppose γ is a set of premises that describe a system Σ . We say that γ satisfies the *Formal Security Condition* (FSC) with respect to L if and only if, for every $b \in O[L]$, the formula

$$\Box(Pr_L(L' = b) = r \rightarrow K_L(Pr_L(L' = b) = r))$$

is derivable from γ .

We say that γ satisfies the *Semantic Interpretation of the FSC with respect to L* if and only if, for every $b \in O[L]$,

$$\gamma \models \Box(Pr_L(L' = b) = r \rightarrow K_L(Pr_L(L' = b) = r))$$

□

Intuitively, FSC says that at all times the low environment knows the probability distribution on its next output.

5.1 Relationship to Probabilistic Noninterference

In this subsection we recall the definition of Probabilistic Noninterference (PNI), as given in [Gra92], and prove that the semantic interpretation of FSC is equivalent to PNI.

Definition 5.2 Let \mathcal{A}_1 and \mathcal{A}_2 be two adversaries that satisfy the Secure Environment Assumption. We will say that \mathcal{A}_1 and \mathcal{A}_2 agree on L behavior iff there exist $\mathcal{H}_1, \mathcal{H}_2$, and \mathcal{L} such that \mathcal{H}_1 and \mathcal{L} are the unique probability functions that describe \mathcal{A}_1 (as in Definition 2.2) and \mathcal{H}_2 and \mathcal{L} are the unique probability functions that describe \mathcal{A}_2 . □

Now, we state the definition of PNI in terms of our model.

Definition 5.3 Let Σ be a system with computation trees $\mathcal{T}(\Sigma)$. We say that Σ *satisfies Probabilistic Non-interference (PNI) with respect to a subject* $L \subseteq C$ iff for any two trees satisfying the Secure Environment Assumption, $T_A, T_{A'} \in \mathcal{T}(\Sigma)$ and any L -event, e , if A and A' agree on L behavior, then

$$\mu_A(e) = \mu_{A'}(e)$$

□

PNI is equivalent to Browne's (independently developed) *Stochastic Non-Interference* [Bro89]. The significance of PNI is that it is arguably a necessary and sufficient condition for a system to be free of covert channels (cf. [Bro91]).

Before we state the main result of this section, we state a lemma that is also interesting in its own right. Space limitations prevent the inclusion of proofs of our results here.

Lemma 5.4 Suppose that T_A and $T_{A'}$ are two trees that agree on L behavior (and satisfy the Secure Environment Assumption). Further suppose that for any two points, $P_1 \in T_A$, $P_2 \in T_{A'}$, and any low output vector, $b \in O[L]$, if P_1 and P_2 have the same L -history, then

$$v_{P_1}(Pr_L(L'_{out} = b)) = v_{P_2}(Pr_L(L'_{out} = b))$$

Then, for any L -event, e ,

$$\mu_A(e_A) = \mu_{A'}(e_{A'})$$

□

We can now state the following theorem relating PNI and FSC.

Theorem 5.5 Let γ be a set of formulae describing Σ and let $L \subseteq C$ be a subject. Then, Σ satisfies PNI with respect to L iff γ satisfies the semantic interpretation of FSC with respect to L . □

A significance of this theorem is that (given soundness as proven in [GS95]) verifying that a system satisfies FSC is equivalent to showing that it satisfies PNI, which (as was previously mentioned) is a necessary and sufficient condition for a system to be free of covert channels.

5.2 Examples, continued

We note here that the security of the encryption box of Example 3.1 with respect to a subject $L \subseteq C$ is formally derivable. In fact, once the assumptions are written down, there is virtually nothing to prove. Recall the system specification: If $C = \{h, l\}$, $I = \{0, 1\}$, and

$O = \{0, 1\}$, then, the system is specified by the following formula.

$$\Box (Pr_C(l'_{out} = 0) = Pr_C(l'_{out} = 1) = 0.5)$$

Recall also that subjects are assumed to always know that the system description holds at all times. Thus,

$$\gamma = \{\Box K_L \Box (Pr_L(L'_{out} = 0) = Pr_L(L'_{out} = 1) = 0.5)\}$$

The only $b \in O[L]$ are O and 1 ; hence, FSC with respect to L for this system is:

$$\Box (Pr_C(L'_{out} = 0) = 0.5 \wedge Pr_C(L'_{out} = 1) = 0.5) \rightarrow K_L (Pr_L(L'_{out} = 0) = 0.5 \wedge Pr_L(L'_{out} = 1) = 0.5)$$

But, this is trivially derivable from γ .

We also observe that for the insecure encryption box of Example 3.2, $\not\models$ FSC (where γ encompasses those formulae that embody the system description and our assumptions about knowledge thereof). It is obvious that the insecure encryption box fails to satisfy PNI. By the attack described in the original example, we can easily find two adversaries that satisfy the Secure Environment Assumption and agree on low behavior and yet disagree on the probability of certain low events. Indeed, the low environment can assign 0/1 probabilities to any output sent by the high part of the adversary. By theorem 5.5, we thus have that $\gamma \not\models$ SSC. And, by soundness, it follows that $\gamma \not\models$ FSC.

6 Conclusions and Relation to Previous Results

In [GS92] a definition of security was presented that we called the *Syntactic Security Condition* (SSC). In [GS95] this was rendered into the framework of this paper using the powerful-subject-knowledge and permitted-knowledge operators of our logic. This definition is based on the definition of "Causality" given by Bieber and Cuppens [BC92], which was based on the work of Glasgow, MacEwen, and Panangaden [GMP90]. SSC says that a system is secure with respect to the set of low processes, L , if and only if, for any logical formula φ , the following formula is derivable from the given premises describing the behavior of the system Σ .

$$\Box (K_L(\varphi) \rightarrow R_L(\varphi)) \quad (1)$$

Although the statement of SSC is almost syntactically identical to Bieber and Cuppens' definition of Causality, due to the differences in the semantics of the respective logics, the meanings of SSC and Causality are different.⁴ In fact, it is straightforward to show that for deterministic systems, the meaning of SSC is equivalent

⁴For technical reasons, Bieber and Cuppens' definition omitted the \Box operator.

to the meaning of Causality. Thus, since SSC additionally applies to probabilistic systems, SSC can be viewed as a generalization of Causality. However, since SSC requires that we have a derivation for all formulae of the language, even those having nothing to do with the system, it is of limited value for verification.

This led us to develop a syntactic verification condition. Though the syntactic verification condition we originally gave in [GS92] appears somewhat complex, the formula given in [GS95], hereafter called ‘SVC’, is almost the same as FSC. The only difference is that SVC has a subscript C in the antecedent where FSC has a subscript L , i.e., $\Box(Pr_C(L' = b) = r \rightarrow K_L(Pr_L(L' = b) = r))$ instead of $\Box(Pr_L(L' = b) = r \rightarrow K_L(Pr_L(L' = b) = r))$.

Though subtle the difference is important, primarily because SVC is too strong. Recall Example 3.1, the secure encryption box. In that example the exclusive-or of each high input bit was taken with a bit of key stream that was equally likely to be 0 or a 1. The result was then output to low at the next clock tick. Consider the following variation on this. The result of the XOR is output to high rather than to low one tick after each high bit is input. The same value is then output to low on the next tick. (This might be done for auditing purposes. In this sense the example is reminiscent of one given in [McL90].) It should be readily apparent that the encryption box in this example is still secure. It should also be readily apparent that this example violates SVC but not FSC. Thus SVC is too strong a criterion for multilevel security. Whether FSC or SVC is easier to verify is impossible to say without further practical examination. Our examples above are too trivial to be taken as representative. As of this writing we are still examining these and other verification conditions for their practical significance. Regardless of which condition, if any, ultimately proves to be practically useful, FSC remains of theoretical importance: its meaning is equivalent to PNI, and, unlike SSC, it is in principal syntactically verifiable.

SVC also remains important, for tying logical characterizations of security to information-theoretic characterizations. The same is true for SSC, perhaps all the more so because it is itself a version of a previous characterization of security [GMP90, BC92]. However, in order to provide that connection in the case of SSC we were forced to represent the somewhat unusual modalities of strong-subject knowledge and permitted knowledge.⁵ And, this required the development of rather complex accessibility relations to capture them

semantically. While these are revealing and interesting in their own right, FSC requires only a standard knowledge operator with a standard semantics. That is one more advantage to this characterization.

Finally, we note that it only became apparent to us that we could effectively capture PNI using FSC after reformulating the earlier work of [GS92] in the framework presented here. This provides further evidence that the framework introduced herein is a useful one.

Acknowledgements

We thank John McLean for helpful discussions and the anonymous referees for many helpful comments.

References

- [Aba90] Martín Abadi. An axiomatization of Lamport’s temporal logic of actions. Technical Report 65, SRC, October 1990.
- [BC92] P. Bieber and F. Cuppens. A logical view of secure dependencies. *Journal of Computer Security*, 1(1):99–129, 1992.
- [Bro89] Randy Browne. *Stochastic Non-Interference: Temporal Processes Without Covert Channels*. Odyssey Research Associates, Ithaca, NY, November 1989. unpublished manuscript.
- [Bro91] Randy Browne. The Turing Test and non-information flow. In *Proc. 1991 IEEE Symposium on Research in Security and Privacy*, Oakland, CA, May 1991.
- [Che80] Brian F. Chellas. *Modal Logic: An Introduction*. Cambridge University Press, Cambridge, 1980.
- [Den82] Dorothy E. Denning. *Cryptography and Data Security*. Addison-Wesley, Reading, Massachusetts, 1982.
- [GMP90] Janice Glasgow, Glenn MacEwen, and Prakash Panangaden. A logic for reasoning about security. In *Proc. Computer Security Foundations Workshop III*, Franconia, NH, June 1990.
- [Gol92] Robert Goldblatt. *Logics of Time and Computation*, 2nd edition, volume 7 of *CSLI Lecture Notes*. CSLI Publications, Stanford, California, 1992.
- [Gra92] James W. Gray, III. Toward a mathematical foundation for information flow security. *Journal of Computer Security*, 1(3):255–294, 1992. A preliminary version appears in *Proc. 1991 IEEE Symposium on Research in Security and Privacy*, Oakland, CA, May, 1991.

⁵That the meaning of SSC is equivalent to PNI was first presented in [GS92]. Proof of this result with respect to the logical framework of this paper is given in [GS95]. Similarly, the connection between SVC and an information-theoretic condition (AFM) was first presented in [GS92] and proven in the current framework in [GS95].

- [GS92] James W. Gray, III and Paul F. Syverson. A logical approach to multilevel security of probabilistic systems. In *Proceedings of the 1992 IEEE Computer Society Symposium on Research in Security and Privacy*, pages 164–176, Oakland, CA, May 1992.
- [GS95] James W. Gray, III and Paul F. Syverson. Epistemology of information flow in the multilevel security of probabilistic systems, 1995. Forthcoming technical report.
- [Hal50] Paul R. Halmos. *Measure Theory*. Springer-Verlag, New York, New York, 1950.
- [HT93] Joseph Y. Halpern and Mark R. Tuttle. Knowledge, probability, and adversaries. *JACM*, 40(4):917–962, September 1993.
- [Lam73] Butler W. Lampson. A note on the confinement problem. *Communications of the ACM*, 16(10), October 1973.
- [Lam91] Leslie Lamport. The temporal logic of actions. Technical Report 79, DEC Systems Research Center, Palo Alto, CA, December 1991.
- [Mar74] Jerrold E. Marsden. *Elementary Classical Analysis*. W.H. Freeman, San Francisco, 1974.
- [McC88] Daryl McCullough. Noninterference and the composability of security properties. In *Proceedings of the 1988 IEEE Computer Society Symposium on Security and Privacy*, Oakland, CA, 1988.
- [McC90] Daryl McCullough. A hookup theorem for multilevel security. *IEEE Transactions on Software Engineering*, 16(6):563–568, June 1990.
- [McL90] John McLean. Security models and information flow. In *Proc. 1990 IEEE Symposium on Research in Security and Privacy*, Oakland, CA, May 1990.
- [Mil90] Jonathan K. Millen. Hookup security for synchronous machines. In *Proceedings of the Computer Security Foundations Workshop III*, Franconia, NH, June 1990.
- [MR88] Leo Marcus and Timothy Redmond. A model-theoretic approach to specifying, verifying, and hooking up security policies. In *Proceeding of the Computer Security Foundations Workshop*, Franconia, NH, 1988.
- [Rud] Walter Rudin. *Principals of Mathematical Analysis*. McGraw-Hill, New York.
- [Sei92] Karen Seidel. Probabilistic communicating processes. Technical report, University of Oxford, Lady Margaret Hall, 1992. PhD thesis.
- [Sha58] Claude E. Shannon. Channels with side information at the transmitter. *IBM Journal of Research and Development*, 2:289–293, October 1958. Republished in: David Slepian (ed.), “Key Papers in the Development of Information Theory”, IEEE Press, 1974.
- [Shi84] A. N. Shiriyayev. *Probability*, volume 95 of *Graduate Texts in Mathematics*. Springer-Verlag, 1984.
- [Sut86] David Sutherland. A model of information. In *Proceeding of the 9th National Computer Security Conference*, Baltimore, MD, September 1986.
- [WJ90] J. Todd Wittbold and Dale M. Johnson. Information flow in nondeterministic systems. In *Proceedings of the 1990 IEEE Computer Society Symposium on Research in Security and Privacy*, Oakland, CA, 1990.
- [Wra92] John C. Wray. An analysis of covert timing channels. *The Journal of Computer Security*, 1(3):219–232, 1992.